**190410**

\*Global Variable, Static Variable, Internal Linkage, External Linkage

**Internal Linkage(내부 연결) :** 변수를 선언 했는데 해당 cpp파일 안 어디서든지 사용 가능한 변수

**External Linkage(외부 연결) :** 파일들이 여러 개가 있는데 한 cpp파일에서 선언한 변수를 다른 cpp파일에서도 사용 가능한 변수

**Static변수** : (OS로부터 받은 메모리가 정적이다. 정적으로 같은 메모리 주소를 계속 재사용 한다. 라는 의미로 이해하자.) 선언과 동시에 초기화를 꼭 해줘야 한다.

Q. 전역변수가 위험한 이유

A. 전역변수를 바꾸는 곳이 다른 cpp파일일 경우 의도한대로 코딩이 안되고 디버깅도 힘들다.

Q. 전역변수 그러면 어떻게 해야 될까?

대안 1. Local Variable을 함수의 Parameter로 넘기면서 사용한다.

대안 2. 사용하되 변수명을 g\_(이름) 같은 식으로 지어 전역변수임을 명시해 둔다.

**결국은, 사용 안 하는게 제일 좋다.**

\*External Linkage(외부 연결)

(1) 외부 cpp파일에 있는 함수를 연결시켜주는 방법

대안 1. 메인함수가 있는 cpp파일에 #include “test.cpp” 해서 cpp파일 자체를 포함 시켜준다(잘 사용 안함)

대안 2. 함수의 헤드 부분을 메인함수가 있는 cpp파일에 선언 시켜준다. (Forward Declaration, 함수를 Forward Declaration할 때 앞에 extern은 생략되어 있다고 본다. 원래는 extern void doSomething(); 이다.)

Extern의 의미 : 프로젝트 내의 모든 CPP파일에서 접근이 가능하다.

주의해야 할 예제 1)

①extern int a;

②

③int main()

④{

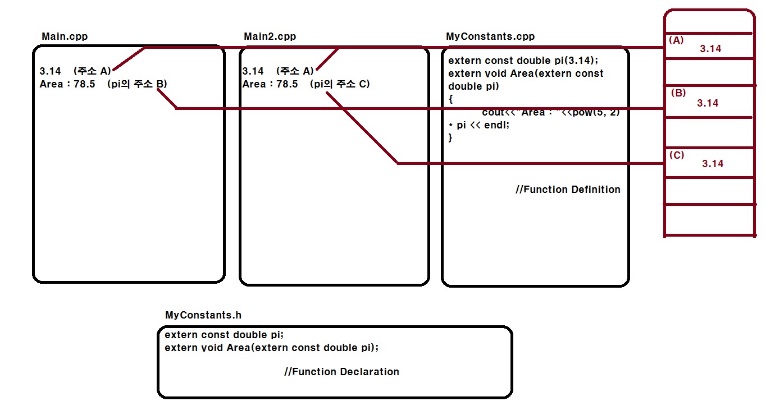
⑤ Cout<<a<<endl;

⑥}

//위 코드의 문제를 Linking Error라고 한다. 컴파일러가 1번문장을 만났을 때 extern을 보고 일단 넘긴다. 그러다가 Linking을 하려고 드니까 모든 OBJ 파일들을 다 찾아도 a의 몸체를 찾을 수 없다. 즉, 초기화를 안 해 메모리를 안 갖고 있어서 에러가 난 것이다.

주의해야 할 예제2)

전역 상수를 헤더파일로 따로 만들어 관리 할 경우 extern 명시를 해주지 않으면 메모리가 낭비 될 수 있다. 실제로 서로 다른 cpp 파일에서 주소 값을 찍어보면 다르다는 걸 알 수 있다.



※단, 함수의 Argument로 이용 할 경우 Argument를 extern 으로 준다고 해도 어쩔 수 없이 메모리 공간을 따로 쓸 수 밖에 없게 된다.